Version 2.2

# Swish Merchant

## Integration Guide

2020-03-24

# Table of contents


**1. Introduction ..... 4**
**1.1 Terms and definitions ..... 4**
**1.2 Document purpose ..... 5**
**1.3 Swish overview ..... 5**
**1.4 Security ..... 5**

**2. Setup ..... 6**
**2.1 Applying for Swish Commerce ..... 6**
**2.2 Technical Integration ..... 7**
2.2.1 Technical Requirements ..... 7
2.2.2 Integration procedure ..... 7
**2.3 Managing certificates ..... 9**
**2.4 Revoking a certificate ..... 9**
**2.5 Termination of Swish Commerce ..... 9**

**3. Payments ..... 10**
**3.1 Overview ..... 10**
**3.2 Payment Requests ..... 10**
3.2.1 Creating a Payment Request ..... 10
3.2.2 M-Commerce Payment Requests ..... 10
3.2.3 E-Commerce Payment Requests ..... 12
3.2.4 Callback ..... 13
**3.3 Payment Refunds ..... 14**
3.3.1 Overview ..... 14
3.3.2 Creating a Refund ..... 14
3.3.3 Callback ..... 16
**3.4 Payouts ..... 16**
3.4.1 Overview ..... 16
3.4.2 Creating a Payout ..... 16

**4. Swish app integration ..... 18**
**4.1 Checking if the Swish app is installed ..... 18**
4.1.1 iOS (Swift) ..... 18
4.1.2 Android (Java) ..... 18
**4.2 Switching to the Swish app ..... 19**
4.2.1 iOS (Swift) ..... 20
4.2.2 Android (Java) ..... 21
4.2.3 JavaScript ..... 22

**5. Swish API ..... 23**
**5.1 Guidelines for using the Swish API ..... 23**
5.1.1 The consumer should be in control of payment requests ..... 23
5.1.2 Use the callback when creating resources ..... 23
5.1.3 Refund transactions – avoid large batches ..... 23
5.1.4 Renewal of the client TLS Certificate ..... 23
5.1.5 Displaying the Swish alias to consumers ..... 23
**5.2 Versions ..... 24**
**5.3 Environments ..... 24**
5.3.1 Test Environment ..... 24
5.3.2 Production Environment ..... 24

**5.4 API Description ..... 25**
5.4.1 Create Payment Request ..... 25
5.4.2 Retrieve Payment Request ..... 27
5.4.3 Cancel payment request ..... 27
5.4.4 Create Refund ..... 29
5.4.5 Retrieve Refund ..... 30
5.4.6 Create Payout ..... 31
5.4.7 Retrieve Payout ..... 33
**5.5 API Objects ..... 35**
5.5.1 Date format ..... 35
5.5.2 Payment Request ..... 35
5.5.3 Refund ..... 36
5.5.4 Create Payout Request ..... 38
5.5.5 Payout ..... 38
5.5.6 Operation ..... 40
5.5.7 Error ..... 40
**6. Support ..... 41**
**6.1 Deployment support ..... 41**
**6.2 Operational status information ..... 41**

# 1. Introduction

## 1.1 Terms and definitions

| Term | Definition |
|---|---|
| Partner | A partner is a company, working with technical integrations, app development, platform development and/or payment services that may help and facilitate merchant integration and operation for Swish.<br>Banks have agreements with the merchants who in turn may have an agreement with a partner. |
| Merchant | A merchant is a company, association or organization which receives payments via Swish.<br>Merchants sign Swish agreements with their respective bank. |
| Merchant Swish Simulator | The Merchant Swish Simulator is a test tool to test the Swish-API. |
| Consumer | A consumer is a private Swish customer that can use the Swish app on a mobile device. |
| CPOC | Certificate Point of Contact – person assigned by the company to manage the certificates |
| Swish Commerce (Swish handel) | Swish Commerce gives the merchants the possibility to use Swish as a payment method in m- and e-commerce. The service is aimed primarily for m- and e-commerce stores, via apps and browsers.<br>Swish Commerce consist of two different payment solutions; Swish m-commerce and Swish e-commerce, a security solution and a function for refunds. All of them are reachable for the merchants through the Swish API.<br>The service can be offered by the banks under a different product name than Swish Commerce. |
| Swish m-commerce | Swish payments from a mobile device made either through an app or via a mobile browser on the same mobile device. |
| Swish e-commerce | Swish payments initiated by the consumer in a browser in equipment other than the mobile device that hosts the Swish app. |
| Swish customer | This is any customer to Swish, either a consumer (person) or a merchant. |
| Payee | This is the Swish customer that receives the payment |
| Payer | This is the Swish customer that makes the payment |
| Alias | A unique identifier for a Swish customer. For a consumer it is the mobile number and for a merchant it is the Swish number. |
| Payment request | A payment request is a transaction sent from a merchant to the Swish system to initiate an e-commerce or m-commerce payment. |
| Payee Payment Reference | A payee payment reference is the merchant's own identifier of the transaction/order to be paid. It is sent to the Swish system as a parameter to the payment request and is later returned in confirmation messages. |
| Refund | A refund is a transaction sent from the merchant to the Swish system to return the whole amount or part of a payment. The reference to the original payment must be provided. |

## 1.2 Document purpose

The integration guide is for anyone who wishes to understand and implement Swish Commerce in their services and systems. The integration guide explains how to connect to the Swish Commerce API and includes information about the payment and refund options related to the Swish Commerce service. More information about the service can be found at https://developer.getswish.se/merchants/.

## 1.3 Swish overview

By enrolling to the Swish Commerce service at the merchant's bank and getting access to the Swish API, merchants can handle payments in e-commerce and m-commerce scenarios in a way which is very convenient and familiar to millions of Swedish consumers. The service builds on the ease-of-use of the person-to-person payment service. When enrolled to the service the merchant can receive payments from all private persons using Swish.

It is also possible for merchants to make refunds in real time using the Swish API. Some banks will also provide the possibility to initiate refunds from the bank's digital channels.

In brief, a payment involves the following steps:

- The merchant creates a payment request using the Swish API that the consumer views and accepts in the Swish app.
- The consumer and the merchant receive payment confirmations immediately when the amount has been transferred from the consumer's to the merchant's account. For security reasons the payment request is only valid during a limited period time for the consumer in the Swish app.

When enrolling to the service, the merchant obtains a Swish alias to one of the merchant's bank accounts. The merchant will also authorize Certificate Point of Contact persons during enrollment. These persons will use the Swish Certificate Management System to manage digital certificates, which is one component of securing the access to the API.

When a payment is made using Swish, the business transaction is between the merchant and the consumer. This transaction implies that the consumer makes an advance payment for purchased goods or services.

## 1.4 Security

In order to protect the Swish API and to ensure the identity of the parties, the security solution encrypts the traffic and authenticates the identities of the merchant and Swish server.

The security solution is implemented as PKI based TLS client/server certificates, where the certificates are issued upon order by the merchant or someone appointed by the merchant. A certificate is valid for 2 years.

A merchant appoints up to 5 persons via their bank, who will be able to log on via BankID/BxID on card or Mobile BankID to the Swish Certificate Management system connected to the security solution. An appointed person can administer their certificates using the system. It includes the possibility to order new certificates and to view, download or revoke current certificates.

After the enrolment process the corporate customer needs to login to the Swish Certificate System to create two different client certificates. One of these certificates is to be installed on their software system and should be used as client certificate to secure the communication between the customer's software system and Swish (here after referred to as *TLS-certificate*). The other one will be used for signing purposes described further down in this document (here after referred to as *signing-certificate*).

# 2. Setup

## 2.1 Applying for Swish Commerce

This user story provides a high-level description of how a merchant applies for the Swish Commerce service.

1. The merchant contacts a bank connected to Swish in order to sign an agreement for the service.
2. The merchant confirms the business terms and signs the agreement with the bank.
    a. The bank obtains and registers the necessary merchant information, including info about the appointed recipients of the API certificate - CPOC (Certificate Point Of Contact). The following personal information is mandatory about the CPOC: Social Security Number, Name, Company Registration Number. Some banks might also require additional information such as e-mail and phone number.
    b. A Swish number is created for the agreement.
    c. The bank sends an enrollment request to Swish security solution.
3. The security solution receives and registers info about the CPOC connected to the Swish number. The CPOC's are granted access to the certificate management system in the Swish security solution.

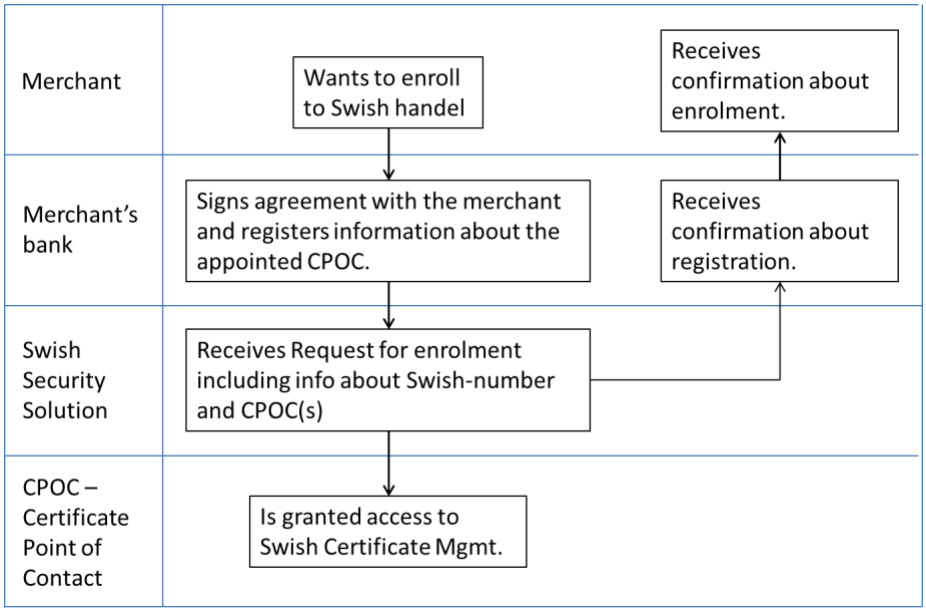



Figure 1: The Swish Commerce application procedure steps 1-3

4. The merchant is now ready for Swish API access.
5. The merchant or the partner needs to generate a CSR-file (Certificate Signing Request). This is normally done by the CPOC.
6. The CPOC logs in to the certificate management system using Mobile BankID, BankID on card or BxID and creates the certificate.
7. The CPOC installs the certificate in the merchant's server and connects it to Swish API.

8. The merchant verifies the connection.

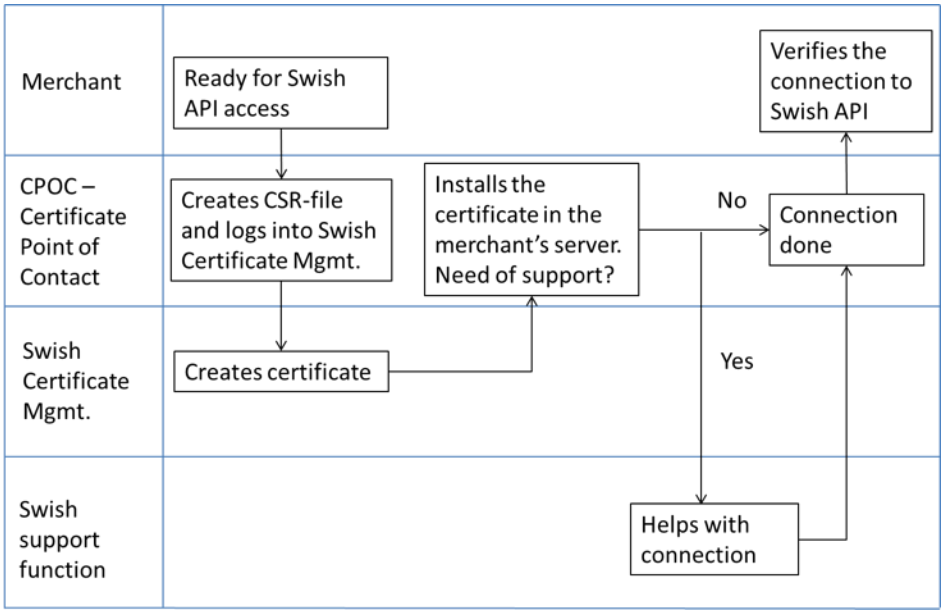



Figure 2: The swish Commerce application procedure steps 4-8

# 2.2 Technical Integration

## 2.2.1 Technical Requirements

The Swish server requires TLS 1.1 or higher.

The merchant must be able to receive the callback HTTPS POST request from the Swish server over TLS. The callback endpoint has to use HTTPS on port 443 and it is highly recommended to use IP filtering as well. For the callback, Swish will be acting as client and the merchant server is acting as server. Swish will validate the merchant callback server TLS certificate against a list of commonly recognized CAs.

For now, the Swish API does not support Server Name Indication (SNI) for the callback functionality.

## 2.2.2 Integration procedure

In order to integrate a merchant commerce solution with the Swish Commerce API, the merchant needs to get a client TLS certificate from Swish Certificate Management and install it on their server. The certificate will be used for client authentication of TLS communication with the Swish API. The following steps needs to be performed:

1. Generate a pair of 4096 bits RSA keys on your server and create a certificate request (CSR) in a PKCS#10 format.
   This step depends on the type of web server solution that is used and differs between different types of servers. The keys are usually generated to a so-called *keystore* (e.g. Java keystore, Microsoft Windows keystore) or file (e.g. openSSL on Apache/Tomcat). For details please consult your web solution documentation or your supplier.

   Note: The following examples are to be considered regarding secure handling of cryptographic keys and certificates. The Customer's keys should be installed by the Customer in secure cryptographic units or should be protected in a similar manner. The keys should only be installed on units necessary for production and back-up purposes. The keys should be deleted at all instances when no longer operational. The keys should at all times be stored with strong encryption and protected

using passwords or more secure procedures, e.g. smart-cards. Passwords used to protect the keys should be handled two jointly and are to be stored in a secure manner so they cannot be lost or subjected to unauthorized access.

It is highly important to protect the private key from unauthorized access. It is recommended to protect the keys with a password if your server provides this option. Care should be taken to protect the passwords as well.

There are no requirements on the content of the CSR (names or other parameters), except for the keys that need to be 4096-bit RSA.

It is possible to install the same certificate on several servers (depending on technical server setup, but no license limitations), or to issue one key pair and certificate per server.

2. Log in to Swish Certificate Management at https://comcert.getswish.net by using Mobile BankID, BankID on card or BxID. Only the person(s) registered by the bank for a specific merchant will be able to perform this step.

3. Provide the organizational number of the merchant and the Swish number for which a certificate is to be generated.

4. Select the "New certificate" tab and paste the content of the generated CSR into the text field. Choose whether the certificate should be in PKCS#7 or PEM format. Consult your documentation regarding which format suits your solution.

5. A new certificate is generated and provided on the screen. Copy the text string and save it to a file. The response (PKCS#7 or PEM) will contain your client certificate and all CA certificates up to the Swish root.

6. Import the generated certificate and all CA certificates to your server. For details on how to perform this step consult your web solution documentation or your supplier.

7. The Swish server is set up with a TLS server certificate, which needs to be verified when initiating TLS from your web server to Swish. Choose to trust DigiCert Global Root CA which can be downloaded here https://www.digicert.com/digicert-root-certificates.htm . For details on how to perform this step consult your web solution documentation or your supplier.

After performing these steps, you should be able to set up TLS with the Swish API.

Note: It is necessary provide the generated certificate together with all CA certificates up to the Swish Root CA in order to correctly set up a TLS session with the Swish API.

Note: No error messages will be returned before a TLS session is successfully established with the Swish API. This means that if the wrong certificate has been used, if the validity time of the certificate has expired, or if the certificate has been revoked, no indication of this is given.

Note: It is recommended to require verification of the Swish API TLS certificate and not to ignore this verification, in case your server allows you to disable server certificate verification.

## 2.3 Managing certificates

Log in to Swish Certificate Management at https://comcert.getswish.net by using Mobile BankID, BankID on card or BxID. Only the person(s) registered by the bank for a specific merchant will be able to perform this step.

Provide the organizational number of the merchant and the Swish number for which a certificate is to be managed.

After logging in a list is provided with all certificates associated with the specific merchant and Swish number, and the status of them. By clicking on “Download” it is possible to see further details and to attain the certificate again.

## 2.4 Revoking a certificate

If the integrity of the merchant’s private key has been compromised, if a certificate has been replaced by a new one, if the service has been terminated, or if the merchant needs to revoke a certificate for some other reason, this can be done via the Swish Certificate Management system.

Log in to Swish Certificate Management at https://comcert.getswish.net by using Mobile BankID, BankID on card or BxID. Only the person(s) registered by the bank for a specific merchant will be able to perform this step.

Provide the organizational number of the merchant and the Swish number for which a certificate is to be revoked.

After logging in a list is provided with all certificates associated with the specific merchant and Swish number, and the status of them. By clicking on the trash can it is possible to revoke a specific certificate.

Please be aware that the certificate is irreversibly revoked and that revoking a certificate that is in use may lead to an interruption of the service.

## 2.5 Termination of Swish Commerce

1. The merchant terminates the agreement with the bank. The service will stop working.
2. The merchant is responsible for termination/revocation of the API certificates.
3. Refunds will not be possible to do on the terminated Swish number.

# 3. Payments

## 3.1 Overview

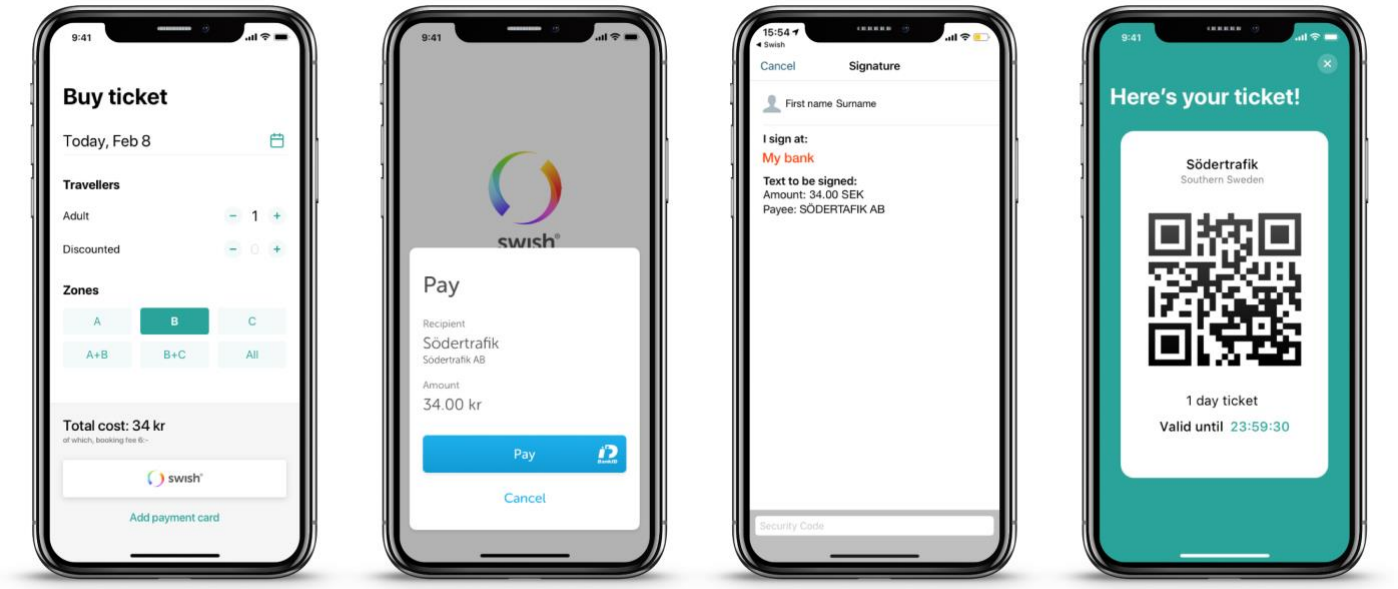
The Swish Commerce API supports two type of payment requests, m-commerce payments and e-commerce payments. It is always the consumer that initiates a payment. When the payment is initiated, a payment request will appear in the Swish app where the consumer can choose to accept or deny the request.

## 3.2 Payment Requests

### 3.2.1 Creating a Payment Request

There are two main flows to this use case, one for Swish m-commerce and one for Swish e-commerce. The main difference is that in the Swish e-commerce case the consumer is prompted for his/her mobile phone number, and then the consumer has to manually open the Swish app. But in the Swish m-commerce case the consumer's mobile phone number is initially not known to the merchant. So instead, in this case, the API returns a Payment request token. This token is used to build a so-called Swish URL, which the merchant can use to call the Swish app from their app. The Payment request token is then a parameter to the Swish URL. Once the payment request has reached a final state (either Paid, Cancelled, Timeout or Error), the merchant provided Callback URL will be called by Swish. Even though this callback contains the payment status information, the merchant server should retrieve the result of the payment request directly from the Swish server (refer to Use the callback for payment requests and refunds for further details).

### 3.2.2 M-Commerce Payment Requests




The consumer chooses to pay with Swish from the merchant's app or website.

The Swish app is activated automatically and preloaded with payment information.

The consumer signs the payment with Mobile BankID.

The consumer gets the payment confirmation in the merchant's app or website.

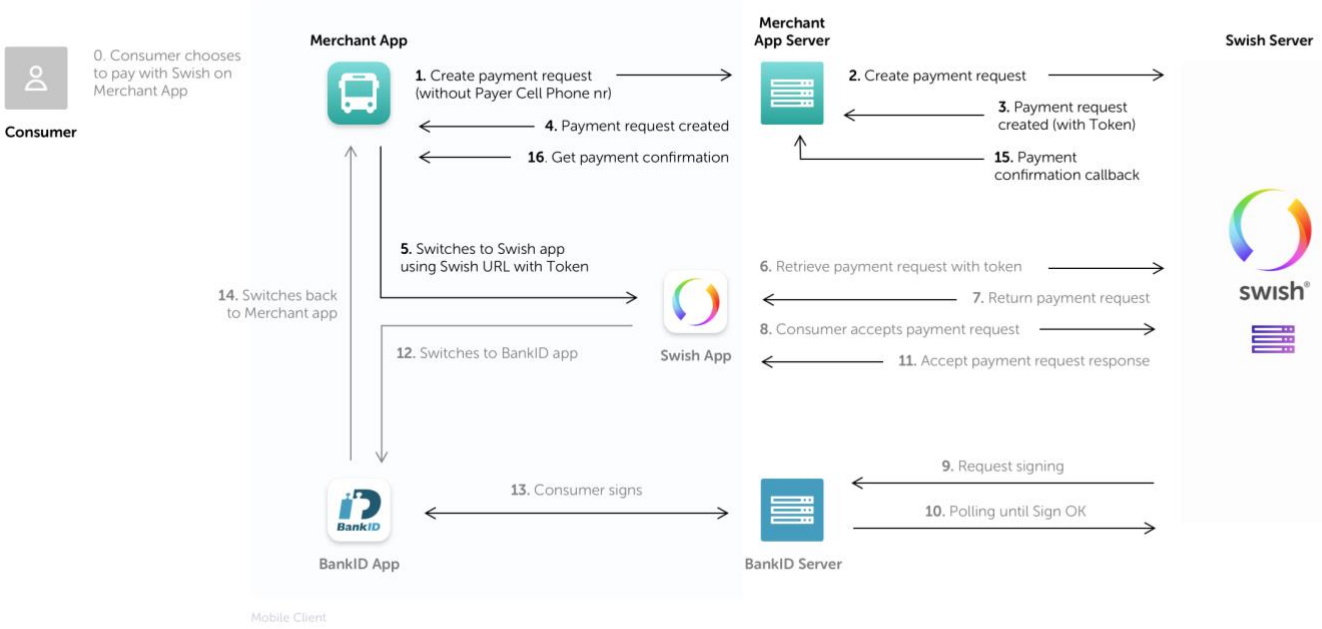
This flow is typically used when the consumer initiates the payment in the merchant's app or website using a mobile device. In this case the consumer does not need to open the Swish-app as the flow switches

automatically between the merchant's app/website and the Swish app. If the payment is completed successfully, it is expected that the merchant's app/website displays a payment confirmation screen as no payment confirmation screen is displayed in the Swish app.

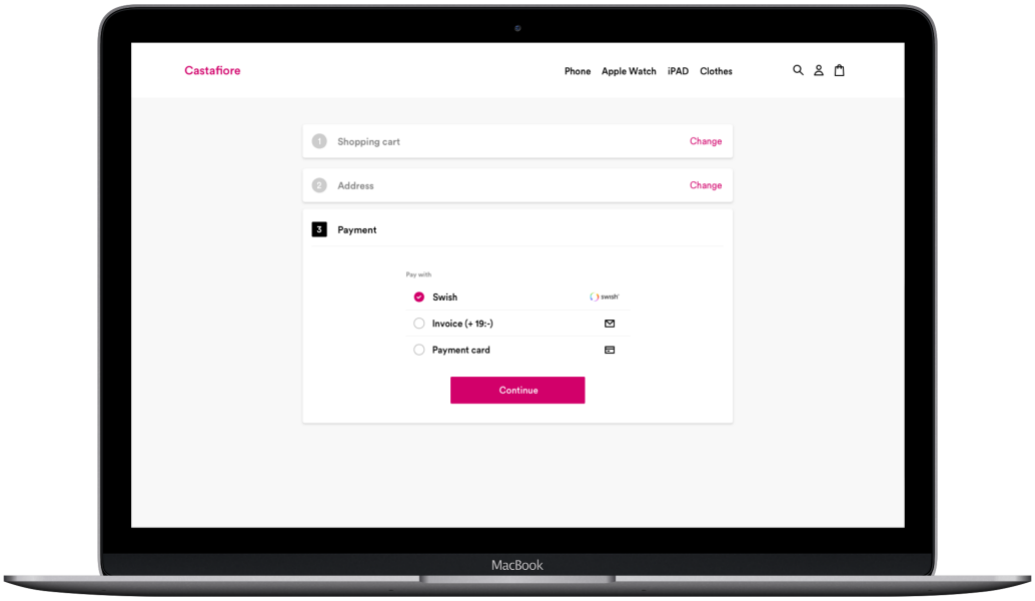
1. The consumer chooses to pay with Swish for a product or a service in the merchant app.
2. The merchant sends a payment request to the Swish system using the API.
    a. The transaction contains data such as: amount, receiving Swish-number, merchant (payee) payment reference and an optional message to the consumer.
3. The merchant receives a Request Token.
4. The consumer's Swish app is opened automatically by the merchant's app/website, showing the payment request that is preloaded with payment information.
    a. The app is opened with the request token as a parameter.
5. The consumer clicks "pay" and the Mobile BankID app opens automatically for signing of the payment transaction.
6. The consumer confirms the payment transaction by signing with Mobile BankID.
7. The amount is transferred in real-time from the consumer's account to the merchant's account.
8. The merchant's app/website is opened again automatically for payment transaction confirmation.
    a. Note: the confirmation screen in Swish-app is not displayed in this flow.
9. The merchant receives a confirmation of successful payment.
10. The consumer can view the payment as a sent payment in the events view in the Swish app.

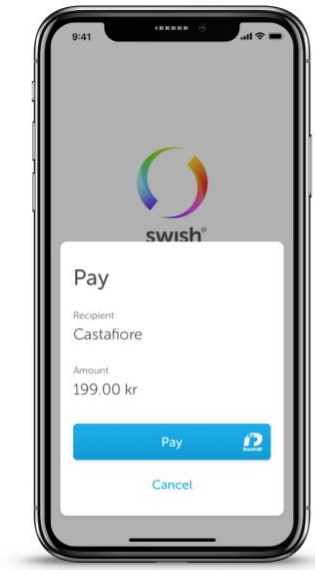
The following diagram shows in detail the steps for creating and accepting an M-Commerce payment request:

**Swish M-Commerce Payment Flow (Happy Case)**

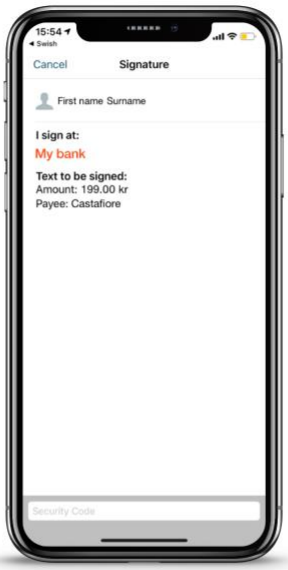


### 3.2.3 E-Commerce Payment Requests

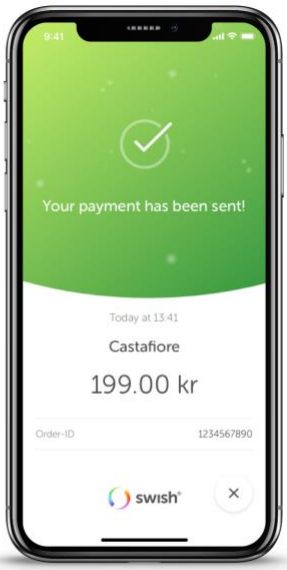



The consumer opens the Swish app, which is preloaded with payment information.




The consumer signs the payment with Mobile BankID.




A payment confirmation is shown inthe Swish app.

This flow is typically used when the consumer initiates the payment in the merchant's website in a desktop environment. In this case the consumer needs to open the Swish-app manually after starting the payment. If the payment is completed successfully, a payment confirmation screen will be displayed in the Swish app.

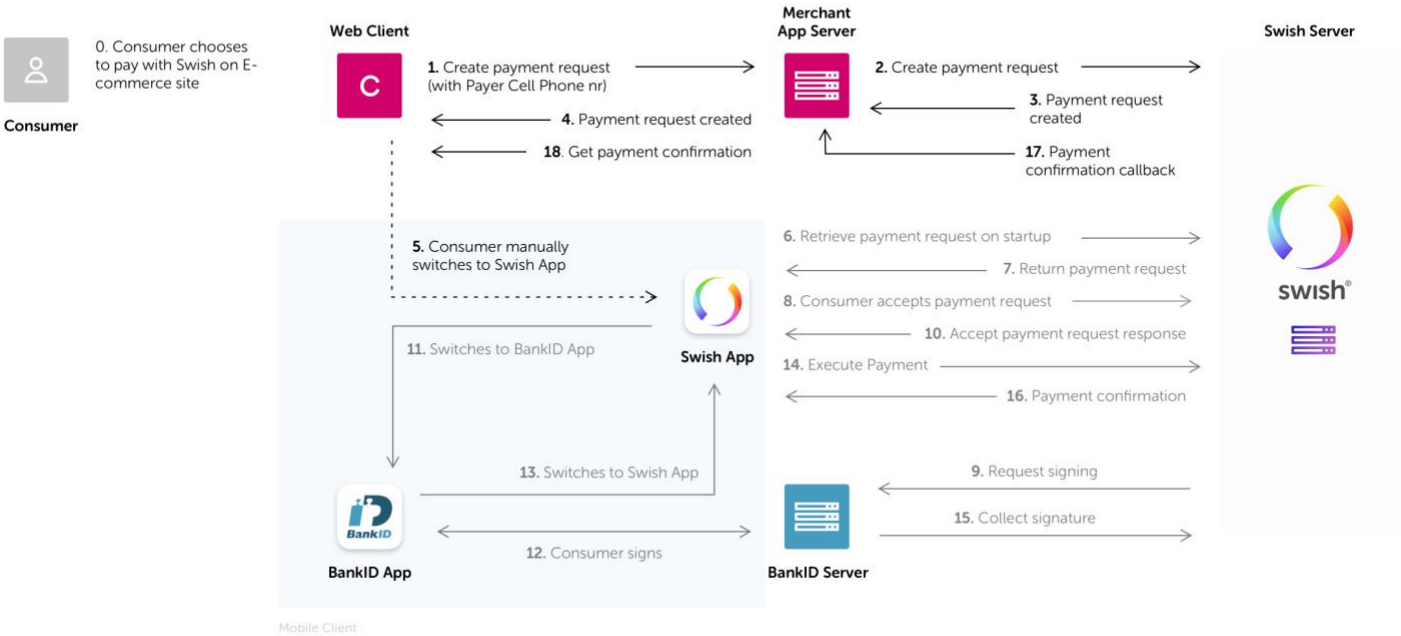
1. The consumer chooses to pay with Swish for a product or a service at the merchant website and enters his/her mobile phone number which is enrolled to Swish.
2. The merchant sends a payment request to the Swish system using the API.

a. The transaction contains data such as: amount, receiving Swish-number, consumer's mobile phone number, merchant payment reference and an optional message to the consumer.

3. The merchant website should inform the consumer to manually open the Swish app to confirm the transaction.
4. The consumer opens the Swish app showing the payment request, which is preloaded with payment information.
5. The consumer clicks "pay" and the Mobile BankID app opens automatically for signing of the payment transaction.
6. The consumer confirms the payment transaction by signing with Mobile BankID.
7. The amount is transferred in real-time from the consumer's account to the merchant's account.
8. The consumer receives a payment confirmation in the Swish app.
9. The merchant receives a confirmation of successful payment.
10. The consumer receives a payment confirmation at the merchant website.
11. The consumer can view the payment as a sent payment in the events view in the Swish app.

The following diagram shows in detail the steps for creating and accepting an E-Commerce payment request.

Swish E-Commerce Payment Flow (Happy Case)




### 3.2.4 Callback

When creating a payment request, a callback URL needs to be specified. Swish will make a callback HTTPS POST request, containing a Payment Request object, to this URL when any of the following events status changes happens:

- PAID - The payment was successful

- DECLINED - The payer declined to make the payment
- ERROR - Some error occurred, like the payment was blocked, payment request timed out etc. See list of error codes for all potential error conditions.
- CANCELLED – The payment request was cancelled either by the merchant or by the payer via the merchant site.

A payment request has to be accepted or declined by the consumer within three (3) minutes for e-commerce and three (3) minutes for m-commerce. When the time has elapsed an ERROR status is returned to the Callback URL. If the consumer accepts the payment request a status is returned to the Callback URL within 12 seconds.

The callback endpoint has to use HTTPS and we highly recommend IP filtering as well. It is however up to the merchant to make sure that the endpoint is available. Swish will only make the callback request once, if the merchant has not received a callback response after the timeout, the merchant can choose to perform the Retrieve Payment Request operation. Swish will always try to make a callback request before the timeout period, but if it times out, then a timeout callback is sent with status ERROR and the error code will have value TM01.

## 3.3 Payment Refunds

### 3.3.1 Overview

A merchant that has received a Swish payment can refund the whole or part of the original transaction amount to the consumer.

A refund can only be done on an existing payment. The number of refunds on one payment is unlimited, until the total amount reaches the amount of the original payment. A payer Order payment reference ID and message to the consumer can be attached to the refund but these are optional. If the refund is successful, a message will be sent to the payee's app.

A refund can be made on a payment for 12 months.

### 3.3.2 Creating a Refund

There are two ways to make a refund: through the bank channel or through the API channel. This section contains a high-level description of a refund through the API channel.

1. As a merchant, I have received a Swish payment and wish for some reason to refund the whole or part of the original amount to the payer.
2. The merchant chooses which payment that is to be completely or partially refunded. The merchant specifies the amount to refund and sends the refund.
3. The merchant will also be able to send its own information that will be shown to the payer in the events view in the Swish app, and also on the payee's bank account statement. The information will also be used by the company for tallying.
4. The merchant receives a confirmation that the refund has been completed.
5. As recipient of a refund, the payee receives a payment notification in the Swish app. The payment is marked as a "Refund" in the events view in the Swish app.

Alternative flow – "payment notification":
In cases when the recipient of a refund cannot receive data push notifications at the moment, the refund will

be visible in the Swish app next time the recipient logs in. The refund will also appear in the recipient's bank account statement.

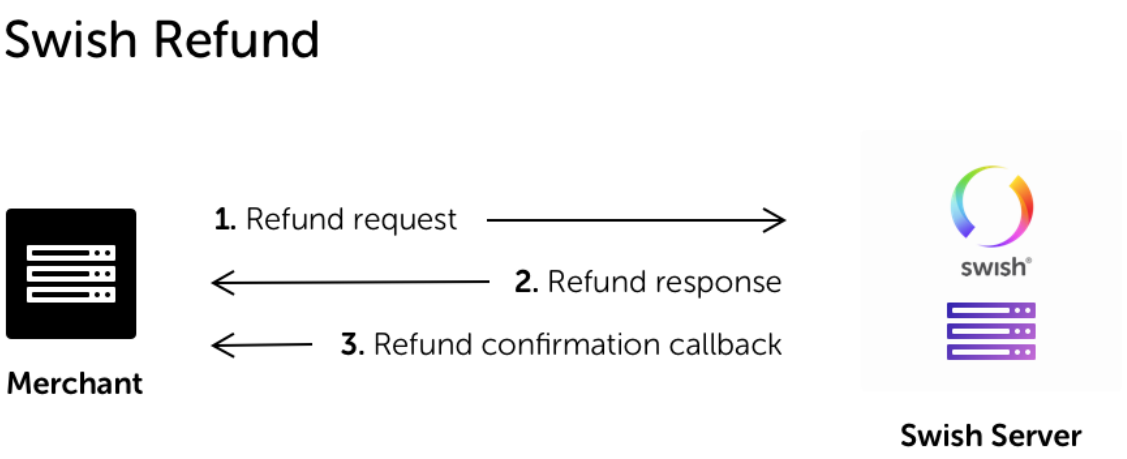
Alternative flow – "refund receiver not connected to Swish":
In cases when the recipient of a refund has terminated the Swish agreement since the original payment occurred, the merchant will receive an error message stating that the refund cannot be processed. The refund must in this case be done via another channel.

Refunds are initiated based on a Payment reference from an earlier payment. To make a refund, perform a Create Refund operation similar to how you create a payment request. The result of the refund is returned in a callback, similar to how a payment request works. A refund normally completes much faster than a payment request, but a callback is used because the actual payment might take a long time. The callback, in the happy case, will return an intermediate response with the status DEBITED. This response is guaranteed to have returned in under 10 seconds or you will get an ERROR response. The DEBITED response means that the money has been taken from the merchants (payers) account but has not been put into the payees account yet. Normally this should happen very soon afterwards, but this "might" take a long time. Moreover, it is not guaranteed to succeed, in other words the receiving bank might refuse to put money into the account. In that case the commerce customer will receive an ERROR response and the money is put back into the commerce customers account. So, these are the potential callback scenarios:

1. Happy case: DEBITED, PAID
2. Early error: ERROR
3. Late error: DEBITED, ERROR

So, in other words there is a tradeoff here, between speed and accuracy that the merchant needs to make:

1. Use the early fast guaranteed response of DEBITED to give a quick response that might turn out to be inaccurate later on.
2. Ignore the DEBITED response and wait for the PAID response that is always accurate but not always fast.



### 3.3.3 Callback

Swish will make a callback HTTPS POST request with a Refund Object to the Callback URL supplied in the Create Refund operation when either of the following status changes happens:

- DEBITED – Money has been withdrawn from your account
- PAID - The payment was successful
- ERROR - Some error occurred. See list of error codes for all potential error conditions.

## 3.4 Payouts

### 3.4.1 Overview

Swish payout is intended to solve the need of making instant payouts from corporations to a large part of the Swedish population. Instead of having to send a file-based payout request taking a couple of days to settle, the company can initiate an instant Swish payout on demand, integrated into the service experience when needed.

### 3.4.2 Creating a Payout

When invoking the payout endpoints, the request body contains three blocks of information, namely payload, signature and callbackUrl.

| payload | signature | callbackUrl |
|---|---|---|

The payload consists of value pairs in JSON format and carries the information about the payout to be executed. In order to authenticate the client and verify the integrity of the data, a signature signed by the *signing-certificate* needs to be supplied. The signature is built by first creating a hash value of the payload using the SHA-512 algorithm and then signing it (the hash value) with the private key of the *signing-certificate*. The serial number of the *signing-certificate* should also be included in the payload in order for the Swish system to validate the integrity of the payload as well as the validity of the certificate.

Payout request payload can be created as follows:

1. Extract the serial number of the *signing-certificate*. The value should be in hexadecimal. Note that the *TLS-certificate* used for communication purposes with Swish system should **not** be used for this purpose. One way to extract the serial number is to use this command:

```
openssl x509 -in YOUR_CERT.pem -serial -noout
```

2. Create a JSON string representing the payout object as follows:

```
{
        "payoutInstructionUUID": "E4D773858AF5459B96ABCA4B9DBFF94D",
        "payerPaymentReference": "payerRef",
        "payerAlias":"1231388446",
        "payeeAlias": "46711111132",
        "payeeSSN": "197709306828",
        "amount": "100.00",
        "currency": "SEK",
        "payoutType": "PAYOUT",
        "message": "Message to the recipient.",
        "instructionDate":"2019-05-05T12:23:23Z",
        "signingCertificateSerialNumber":"7BE0DA9DE336EDCE5FE9AAFEF39248AE"
}
```

For detailed description of these fields, please see Payout below.

3. Hash the payout object created in step 2, using SHA512. Make sure to use UTF-8 encoding when serializing the payload to a byte stream. Here follows an example in Java:

```
String payload = "JSON_PAYLOAD_GOES_HERE";
byte[] msg_bytes = payload.getBytes("UTF-8");
MessageDigest md = MessageDigest.getInstance("SHA-512");
byte[] hash_value = md.digest(msg_bytes);
```

4. Sign the hash code with the RSA algorithm using the private key of the *signing-certificate* and convert the result to Base64. Here follows an example in Java:

```
KeyPair keyPair = getKeyPair(); // getKeyPair is your own function that
                                // returns your key pair for
                                // signing-certificate
Signature sig = Signature.getInstance("NONEwithRSA");
sig.initSign(keyPair.getPrivate());
sig.update(hash_value);
byte[] signatureBytes = sig.sign();
String signature = Base64.getEncoder().encodeToString(signatureBytes);
```

5. Create a JSON payout request object to send to the Swish system. The payout request content shall be constructed in the following format:

```
{
        "payload": {
                "payoutInstructionUUID": "E4D773858AF5459B96ABCA4B9DBFF94D",
                "payerPaymentReference": "payerRef",
                "payerAlias": "1231388446",
                "payeeAlias": "46711111132",
                "payeeSSN": "197709306828",
                "amount": "100.00",
                "currency": "SEK",
                "payoutType": "PAYOUT",
                "message": "Message to the recipient.",
                "instructionDate": "2019 - 05 - 05 T12: 23: 23 Z",
                "signingCertificateSerialNumber":
"7BE0DA9DE336EDCE5FE9AAFEF39248AE"
        },
        "callbackUrl": "https://not.a.real.caller.com/callback",
        "signature":
"a89xIJY8TBwWzKuTh4Qvx6hrUoMDI3/2RooINGmPUNAM0fzfvUFn9RFcvm4z
2WsLuY8xO0F9aioK0MvG2FiYIeDEZjpgAlWyFxJl3R9dvN2lDFo8MMaseQfOK6IuDyIgrRVyTSrdyKKB
GXRcoihX5CN7xQY7zBgl8AtIz9lOQ0o="
}
```

Where the “payload” is created according to step 2 and the “signature” according to step 3 and 4 above. “callbackURL” is the URL which will be called by Swish system when the status of a payout request is changed.

### 3.4.3 Callback

If the callbackUrl is set in the payout request the Swish system will send status information updates back to the client using this callbackUrl. The callbackUrl will be called by the Swish system every time the state of the Payout Request is changed. If callbackUrl is not provided by the client, no callback will be triggered by the Swish system. In both cases, the client may use the GET operation described below in order to fetch the status of a Payout request at any given time.

It is important to note that the Swish system will attempt to call the callbackURL a second time, if the first call is not successfully received by the merchant system. The second call will be triggered one minute after the first call fails.

Clients can check the current status of an initiated payout request by calling the GET method with the *payoutReference/*payoutInstructionUUID received in response to a created payout request. Note that in normal cases the *payoutReference* and payoutInstructUUID fields will have the same value.

# 4. Swish app integration

## 4.1 Checking if the Swish app is installed

Most modern platforms have a way of checking if a certain app is installed on the consumer's device. Here are some examples on how to perform this check.

### 4.1.1 iOS (Swift)

```
enum StringConstants: String {
    case Host = "paymentrequest"
    case SwishUrl = "swish://"
    case MerchantCallbackUrl = "merchant://"
    case Scheme = "swish"
}

func isSwishAppInstalled() -> Bool {
    guard let url = URL(string: StringConstants.SwishUrl.rawValue) else {
        preconditionFailure("Invalid url")
    }
    return UIApplication.shared.canOpenURL(url)
}
```

### 4.1.2 Android (Java)

```
public static boolean isSwishInstalled(Context context) {
    try {
        context.getPackageManager()
                .getPackageInfo("se.bankgirot.swish", 0);
        return true;
    } catch (PackageManager.NameNotFoundException e) {
        // Swish app is not installed
        return false;
    }
}
```

## 4.2 Switching to the Swish app

The merchant apps, including mobile web browsers, will call the Swish app using the custom URL Scheme:

```
swish://paymentrequest?token=<token>&callbackurl=<callbackURL>
```

| Parameter | Description | Required |
|---|---|---|
| token | The payment request token that the merchant has received from the CPC. Example: token=c28a4061470f4af48973bd2a4642b4fa | Yes |
| callbackurl | This callback URL is called after the payment is finished. It should be URL-encoded and can be for example an app URL or a web URL. Example: callbackurl=merchant%253A%252F%252F | Yes |

When the Swish app is finished, it (or the BankID app) will call the provided callback URL. For the merchant app to react on this call, the merchant app needs to register for that URL scheme and provide code for handling the request.

Code snippets describing how to switch to the Swish app as well as information about declaring URL scheme and handling calls to it are provided below for each platform.

Note that the URL Scheme “merchant://” is used in the examples below. This is only an example – each merchant shall use its own unique scheme.

### 4.2.1 iOS (Swift)

The following code can be used to switch to the Swish app from the merchant app.

```
enum StringConstants: String {
    case Host = "paymentrequest"
    case SwishUrl = "swish://"
    case MerchantCallbackUrl = "merchant://"
    case Scheme = "swish"
}

func openSwishAppWithToken(_ token: String) {
    guard isSwishAppInstalled() else {
        // Swish app is not installed, show error
        return
    }

    guard let callback = encodedCallbackUrl() else {
        preconditionFailure("Callback url is required")
    }

    var urlComponents = URLComponents()
    urlComponents.host = StringConstants.Host.rawValue
    urlComponents.scheme = StringConstants.Scheme.rawValue
    urlComponents.queryItems = [URLQueryItem(name: "token", value: token),
                                URLQueryItem(name: "callbackurl", value: callback)]

    guard let url = urlComponents.url else {
        preconditionFailure("Invalid url")
    }

    UIApplication.shared.open(url, options: [:], completionHandler: { (success) in
        if !success {
            // The URL could not be opened, show error
        }
    })
}


func encodedCallbackUrl() -> String? {
    let callback = StringConstants.MerchantCallbackUrl.rawValue
    let disallowedCharacters = NSCharacterSet(charactersIn: "!*'();:@&=+$,/?%#[]")
    let allowedCharacters = disallowedCharacters.inverted
    return callback.addingPercentEncoding(withAllowedCharacters: allowedCharacters)
}
```

The enable the switch back from Swish, the merchant app needs to register a URL scheme. This is done by including a CFBundleURLTypes key in the app's Info.plist. For more information, see Defining a Custom URL Scheme for Your App.

The merchant app must also implement the following function that will be called when the switch back happens:

```
func application(_ application: UIApplication,
                 open url: URL,
                 options: [UIApplicationOpenURLOptionsKey : Any] = [:] ) -> Bool
```

### 4.2.2 Android (Java)

The following code can be used to switch to Swish from the merchant app.

```
public static boolean openSwishWithToken(Context context, String
token, String callBackUrl) {
    if ( token == null
            || token.length() == 0
            || callBackUrl == null
            || callBackUrl.length() == 0
            || context == null) {
        return false;
    }

    // Construct the uri
    // Note that appendQueryParameter takes care of uri encoding
    // the parameters
    Uri url = new Uri.Builder()
            .scheme("swish")
            .authority("paymentrequest")
            .appendQueryParameter("token", token)
            .appendQueryParameter("callbackurl", callBackUrl)
            .build();

    Intent intent = new Intent(Intent.ACTION_VIEW, url);
    intent.setPackage("se.bankgirot.swish");

    try {
        context.startActivity(intent);
    } catch (Exception e){
        // Unable to start Swish
        return false;
    }

    return true;
}
```

The app manifest file is used to register the URL scheme in the merchant app :

```
<intent-filter>
          <action android:name="android.intent.action.VIEW" />
          <category android:name="android.intent.category.DEFAULT" />
          <category android:name="android.intent.category.BROWSABLE" />
          <data android:scheme="merchant" />
</intent-filter>
```

The merchant app also needs to process the intent in onCreate and onNewIntent methods when the switch back happens.

### 4.2.3 JavaScript

The URL syntax below works on most built-in web browsers:

```
window.location =
"swish://paymentrequest?token=c28a4061470f4af48973bd2a4642b4fa&callbac
kurl=merchant%253A%252F%252F";
```

# 5. Swish API

## 5.1 Guidelines for using the Swish API

When integrating with the Swish API it is recommended to adhere to the following guidelines in order to achieve stable performance of the system and a smooth consumer experience.

### 5.1.1 The consumer should be in control of payment requests

Each payment request transaction sent to the API must be initiated by a physical paying consumer. The merchant must make sure that the consumer does not receive what he/she perceives as “spam” or unwanted payment requests.

### 5.1.2 Use the callback when creating resources

When creating a payment request, refund or a payout, a callback is provided to the merchant with the status of the operation. In a typical scenario, this callback should be used for finding out the status of the payment. As a backup there is also a “Retrieve” operation for reconciliation in the case that the normal callback fails for some reason. Note that this is a backup – and should not be the default way for receiving the payment status.

### 5.1.3 Refund transactions – avoid large batches

The “create refund” API is intended for real-time one-by-one calls. It is not intended for batching up a large quantity and then sending the whole batch in a short period of time.

There should be at least 1 second between each refund transaction and if more than 100 transactions are to be sent in a sequence, they should be sent during night time.

### 5.1.4 Renewal of the client TLS Certificate

The validity of the client TLS certificate is two years. It is the merchant's responsibility to generate new keys and certificates in due time, prior to the expiry of the old certificate, in order to ensure uninterrupted functionality of the commerce site. The merchant could authorize another company (a partner to the merchant) to manage the certificate renewal process.

### 5.1.5 Displaying the Swish alias to consumers

When enrolling to Swish Commerce the merchant will receive a Swish alias (123 XXX YYYY) which uniquely identifies the enrolment, and which is used as an alias to the payee’s bank account.

We recommend e-commerce and m-commerce merchants not to expose this to consumers since it:

1. Can be used for unprompted payments by entering the Swish alias in the Swish app.
2. Some banks may block unprompted payments to Swish aliases enrolled to “Swish Commerce”

The Swish alias for transactions generated by payment requests or refunds will not be displayed by the Swish app or the bank’s consumer interfaces.

## 5.2 Versions

Changes may be made to the API to correct errors or to introduce new functionality. When changed, a new version of the API will be made available via a new URL. Merchants should always use the latest version of the API.

The general rule is that old versions of the API will be discontinued two years after the release of the successor. But if deemed necessary, for example for security reasons, a version of the API may be discontinued prematurely. As new functionality is introduced to the system the behavior of an existing version of the API may change, e.g. existing faults may also be used in new situations.

## 5.3 Environments

### 5.3.1 Test Environment

A Merchant Swish Simulator is available for merchants to test their integration with the Swish Commerce API. The Merchant Swish Simulator will validate requests and return simulated but correctly formatted responses. The Merchant Swish Simulator will return a simulated result of the request in the callback URL. It is also possible to retrieve the payment request status and to simulate different error situations.

A user guide for the Merchant Swish Simulator can be found at: https://developer.getswish.se/merchants/.

### 5.3.2 Production Environment

The Swish server IP address for IP filtering:

213.132.115.94:443

Swish API URL:

https://cpc.getswish.net/swish-cpcapi/api/v1/paymentrequests

https://cpc.getswish.net/swish-cpcapi/api/v1/refunds

https://cpc.getswish.net/swish-cpcapi/api/v1/payouts

Swish server TLS certificate is issued under the following root CA that should to be configured as trusted:

CN = DigiCert Global Root CA

OU = www.digicert.com

O = DigiCert Inc

The complete certificate chain of the Swish server TLS certificate is available through Swish Certificate Management.

## 5.4 API Description

### 5.4.1 Create Payment Request

**Request**

```
POST /api/v1/paymentrequests
```

**Parameters**

None

**Request body**

The HTTP request body has to contain a Payment Request object.

For that object,

You must specify a value for these properties:

- callbackUrl
- payeeAlias
- amount
- currency

You may specify a value for these properties:

- payeePaymentReference
- payerAlias
- message

**Response**

If successful, the following response headers are returned:

| Header | Optional | Description |
|---|---|---|
| Location | No | An URL for retrieving the status of the payment request. |
| PaymentRequestToken | Yes | Returned when creating an m-commerce payment request. The token to use when opening the Swish app. |

If a HTTP status code 422 error occurs, an array of Error objects are returned.

**Errors**

| HTTP status code | Description |
|---|---|
| 201 Created | Payment request was successfully created. Will return a Location header and if it is Swish m-commerce case, it will also return PaymentRequestToken header. |
| 400 Bad Request | The Create Payment Request operation was malformed. |
| 401 Unauthorized | There are authentication problems with the certificate. Or the Swish number in the certificate is not enrolled. Will return nothing else. |
| 403 Forbidden | The payeeAlias in the payment request object is not the same as merchant's Swish number. |
| 415 Unsupported Media Type | The Content-Type header is not "application/json". Will return nothing else |
| 422 Unprocessable Entity | There are validation errors. Will return an array of Error objects. |

| | |
|---|---|
| 500 Internal Server Error | There was some unknown/unforeseen error that occurred on the server, this should normally not happen. Will return nothing else |

**Validation errors**

The following validation errors might be returned when the HTTP status code is 422.

| Error code | Description |
|---|---|
| FF08 | PaymentReference is invalid. |
| RP03 | Callback URL is missing or does not use HTTPS. |
| BE18 | Payer alias is invalid. |
| RP01 | Missing Merchant Swish Number. |
| PA02 | Amount value is missing or not a valid number. |
| AM06 | Specified transaction amount is less than agreed minimum. |
| AM02 | Amount value is too large. |
| AM03 | Invalid or missing Currency. |
| RP02 | Wrong formatted message. |
| RP06 | A payment request already exists for that payer. Only applicable for Swish e-commerce. |
| ACMT03 | Payer not Enrolled. |
| ACMT01 | Counterpart is not activated. |
| ACMT07 | Payee not Enrolled. |

**Example request (e-commerce)**

```
curl -v --data '{ "payeePaymentReference": "0123456789", "callbackUrl":
"https://example.com/api/swishcb/paymentrequests", "payerAlias":
"4671234768", "payeeAlias": "1231181189", "amount": "100", "currency":
"SEK", "message": "Kingston USB Flash Drive 8 GB" }' -H "Content-Type:
application/json" POST https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/paymentrequests --cert "Swish Merchant Test Certificate
1231181189.p12:swish" --cert-type p12 --cacert "Swish TLS Root CA.pem"
```

**Example response (e-commerce)**

```
< HTTP/1.1 201
< Location: https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/paymentrequests/DFEC8B87CFC74882BCC832DA6B125332
< Server: nginx/1.12.1
< Connection: keep-alive
< Content-Length: 0
< Date: Fri, 04 Jan 2019 08:28:17 GMT
<
* Connection #1 to host mss.cpc.getswish.net left intact
```

**Example request (m-commerce)**

```
curl -v --data '{ "payeePaymentReference": "0123456789", "callbackUrl":
"https://example.com/api/swishcb/paymentrequests", "payeeAlias":
"1231181189", "amount": "100", "currency": "SEK", "message": "Kingston
USB Flash Drive 8 GB" }' -H "Content-Type: application/json" POST
https://mss.cpc.getswish.net/swish-cpcapi/api/v1/paymentrequests --cert
"Swish Merchant Test Certificate 1231181189.p12:swish" --cert-type p12 --
cacert "Swish TLS Root CA.pem"
```

**Example response (m-commerce)**

```
< HTTP/1.1 201
< Location: https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/paymentrequests/11A86BE70EA346E4B1C39C874173F088
< Server: nginx/1.12.1
< Connection: keep-alive
< PaymentRequestToken: ed16db6f415145ec93642e294c904378
< Content-Length: 0
< Date: Fri, 04 Jan 2019 08:34:59 GMT
<
```

```
* Connection #1 to host mss.cpc.getswish.net left intact
```

### 5.4.2 Retrieve Payment Request

**Request**

```
GET /api/v1/paymentrequests/{id}
```

**Parameters**

| Name | Description |
|---|---|
| id | The identifier of the payment request to retrieve.<br>Example: 11A86BE70EA346E4B1C39C874173F088 |

**Request body**

None

**Response**

If successful, a Payment Request object is returned.

**Errors**

| HTTP status code | Description |
|---|---|
| 200 OK | The Payment request was found. Will return a Payment Request Object. |
| 401 Unauthorized | There are authentication problems with the certificate. Or the Swish number in the certificate is not enrolled. Will return nothing else. |
| 404 Not found | The Payment request was not found, or it was not created by the merchant. Will return nothing else. |
| 500 Internal Server Error | There was some unknown/unforeseen error that occurred on the server, this should normally not happen. Will return nothing else. |

**Example request**

```
curl -v "Content-Type: application/json" GET
https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/paymentrequests/5D59DA1B1632424E874DDB219AD54597 --cert
"Swish Merchant Test Certificate 1231181189.p12:swish" --cert-type p12 --
cacert "Swish TLS Root CA.pem"
```

**Example response**

```
< HTTP/1.1 200
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Fri, 04 Jan 2019 09:00:29 GMT
<
* Connection #1 to host mss.cpc.getswish.net left intact
{"id":"5D59DA1B1632424E874DDB219AD54597","payeePaymentReference":"0123456
789","paymentReference":"1E2FC19E5E5E4E18916609B7F8911C12","callbackUrl":
"https://example.com/api/swishcb/paymentrequests","payerAlias":"467123476
8","payeeAlias":"1231181189","amount":100.00,"currency":"SEK","message":"
Kingston USB Flash Drive 8 GB","status":"PAID","dateCreated":"2019-01-
02T14:29:51.092Z","datePaid":"2019-01-
02T14:29:55.093Z","errorCode":null,"errorMessage":""}
```

### 5.4.3 Cancel payment request

**Request**

```
PATCH /api/v1/paymentrequests/{id}
```

**Parameters**

None

**Request body**

The HTTP request body has to contain a list of Operation objects.

For that object,

You must specify a value for these properties:

- op
- path
- value

Currently, the only supported operation is op = “replace”, path = “/status”, value = “cancelled”.

**Request headers**

| Header | Value |
|---|---|
| Content-Type | application/json-patch+json |

**Response**

If successful a Payment Request object is returned.

**Errors**

| HTTP status code | Description |
|---|---|
| 200 OK | The request completed successfully. Will return a Payment Request Object. |
| 401 Unauthorized | There are authentication problems with the certificate. Or the Swish number in the certificate is not enrolled. Will return nothing else. |
| 404 Not found | The Payment request was not found, or it was not created by the merchant. Will return nothing else. |
| 415 Unsupported Media Type | The MIME type in the Content-Type header is missing or wrong. |
| 422 Unprocessable Entity | The operation could not be performed, either because it is invalid (error code “PA01”) or because it is in a non-cancellable state (error code “RP07”). Will return an array of Error objects. |
| 500 Internal Server Error | There was some unknown/unforeseen error that occurred on the server, this should normally not happen. Will return nothing else. |

**Example request**

```
curl -v --data '[{ "op": "replace", "path": "/status", "value":
"cancelled" }]' -H "Content-Type: application/json-patch+json" --request
PATCH https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/paymentrequests/5D59DA1B1632424E874DDB219AD54597 --cert
"Swish Merchant Test Certificate 1231181189.p12:swish" --cert-type p12 --
cacert "Swish TLS Root CA.pem"
```

**Example response**

```
< HTTP/1.1 200
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Fri, 04 Jan 2019 09:00:29 GMT
<
* Connection #1 to host mss.cpc.getswish.net left intact
{
    "id":"5D59DA1B1632424E874DDB219AD54597",
    "payeePaymentReference":"0123456789",
    "paymentReference":"1E2FC19E5E5E4E18916609B7F8911C12",
    "callbackUrl": "https://example.com/api/swishcb/paymentrequests",
    "payerAlias":"4671234768",
    "payeeAlias":"1231181189",
    "amount":100.00,
    "currency":"SEK",
    "message":"Kingston USB Flash Drive 8 GB",
    "status":"CANCELLED",
    "dateCreated":"2019-04-11T09:58:51.092Z",
```

```
    "datePaid":null,
}
```

### 5.4.4 Create Refund

**Request**

```
POST /api/v1/refunds
```

**Parameters**

None

**Request body**

The HTTP request body has to contain a Refund object.

For that object,

You must specify a value for these properties:

- originalPaymentReference
- callbackUrl
- payerAlias
- amount
- currency

You may specify a value for these properties:

- payerPaymentReference
- messageResponse

If successful, the following response headers are returned:

| Header | Optional | Description |
|---|---|---|
| Location | No | An URL for retrieving the status of the payment request. |

If a HTTP status code 422 error occurs, an array of Error objects are returned.

**Errors**

| HTTP status code | Description |
|---|---|
| 201 Created | The Refund was successfully created. Will return a Location header. |
| 400 Bad Request | The Create refund POST operation was malformed. |
| 401 Unauthorized | There are authentication problems with the certificate. Or the Swish number in the certificate is not enrolled. Will return nothing else. |
| 403 Forbidden | The payerAlias in the refund object is not the same as merchant's Swish number. |
| 415 Unsupported Media Type | The Content-Type header is not "application/json". Will return nothing else. |
| 422 Unprocessable Entity | There are validation errors. Will return an array of Error objects. |
| 500 Internal Server Error | There was some unknown/unforeseen error that occurred on the server, this should normally not happen. Will return nothing else |
| 504 Gateway Timeout | The Bank validation response took too long, and Swish timed out. This rarely happens. |

**Validation errors**

The following validation errors might be returned when the HTTP status code is 422.

| Error code | Description |
|---|---|
| FF08 | PaymentReference is invalid. |
| RP03 | Callback URL is missing or does not use HTTPS. |
| PA02 | Amount value is missing or not a valid number. |
| AM06 | Specified transaction amount is less than agreed minimum. |
| RF08 | Amount value is too large, or amount exceeds the amount of the original payment minus any previous refunds. Note: the remaining available amount is put into the additional information field. |
| AM03 | Invalid or missing Currency. |
| RP01 | Missing Merchant Swish Number. |
| RP02 | Wrong formatted message. |
| ACMT07 | Payee not Enrolled. |
| ACMT01 | Counterpart is not activated. |
| RF02 | Original Payment not found or original payment is more than 13 months old. |
| RF03 | Payer alias in the refund does not match the payee alias in the original payment. |
| RF04 | Payer organization number do not match original payment payee organization number. |
| RF06 | The Payer SSN in the original payment is not the same as the SSN for the current Payee. Note: Typically, this means that the Mobile number has been transferred to another person. |
| RF07 | Transaction declined. |
| FF10 | Bank system processing error. |
| BE18 | Payer alias is invalid. |

**Example request**

```
curl -v --data '{ "originalPaymentReference":
"5D59DA1B1632424E874DDB219AD54597", "callbackUrl":
"https://example.com/api/swishcb/paymentrequests", "payerAlias":
"1231181189", "amount": "100", "currency": "SEK", "message": "Refund for
Kingston USB Flash Drive 8 GB" }' -H "Content-Type: application/json"
POST https://mss.cpc.getswish.net/swish-cpcapi/api/v1/refunds --cert
"Swish Merchant Test Certificate 1231181189.p12:swish" --cert-type p12 --
cacert "Swish TLS Root CA.pem"
```

**Example response**

```
< HTTP/1.1 201
< Location: https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/refunds/2EA344A95DD941D1ACC2F94FBB898180
< Server: nginx/1.12.1
< Connection: keep-alive
< Content-Length: 0
< Date: Fri, 04 Jan 2019 10:29:43 GMT
<
* Connection #1 to host mss.cpc.getswish.net left intact
```

### 5.4.5 Retrieve Refund

**Request**

```
GET /api/v1/refunds/{id}
```

**Parameters**

| Name | Description |
|---|---|
| id | The identifier of the refund to retrieve. |

| | |
|---|---|
| | Example: 2EA344A95DD941D1ACC2F94FBB898180 |

**Request body**

None

**Response**

If successful, a Refund object is returned.

**Errors**

| HTTP status code | Description |
|---|---|
| 200 OK | The refund was found. Will return a Refund object. |
| 401 Unauthorized | There are authentication problems with the certificate. Or the Swish number in the certificate is not enrolled. Will return nothing else. |
| 404 Not found | No refund was found, or it was not created by the merchant. Will return nothing else. |
| 500 Internal Server Error | There was some unknown/unforeseen error that occurred on the server, this should normally not happen. Will return nothing else |

**Example request**

```
curl -v "Content-Type: application/json" GET
https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/refunds/2EA344A95DD941D1ACC2F94FBB898180 --cert "Swish
Merchant Test Certificate 1231181189.p12:swish" --cert-type p12 --cacert
"Swish TLS Root CA.pem"
```

**Example response**

```
< HTTP/1.1 200
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Fri, 04 Jan 2019 12:00:12 GMT
<
* Connection #1 to host mss.cpc.getswish.net left intact
{"id":"2EA344A95DD941D1ACC2F94FBB898180","paymentReference":"9374A9192E73
43F39048E7061DB1DDF3","payerPaymentReference":"","originalPaymentReferenc
e":"5D59DA1B1632424E874DDB219AD54597","callbackUrl":"https://example.com/
api/swishcb/paymentrequests","payerAlias":"1231181189","payeeAlias":null,
"amount":100.00,"currency":"SEK","message":"Refund for Kingston USB Flash
Drive 8 GB","status":"PAID","dateCreated":"2019-01-
04T10:29:43.683Z","datePaid":"2019-01-
04T10:29:52.543Z","errorMessage":null,"additionalInformation":null,"error
Code":null}
```

### 5.4.6 Create Payout

**Request**

```
POST /api/v1/payouts/
```

**Parameters**

None

**Request body**

The HTTP request body has to contain a Create Payout Request object.

For that object,

You must specify a value for these properties:

- payload.payoutInstructionUUID
- payload.payerPaymentReference

- payload.payerAlias
- payload.payeeAlias
- payload.payeeSSN
- payload.amount
- payload.currency
- payload.payoutType
- payload.instructionDate
- payload.signingCertificateSerialNumber
- signature

You may specify a value for these properties:

- callbackUrl
- payload.message

**Response**

If successful, the following response headers are returned:

| Header | Optional | Description |
|---|---|---|
| Location | No | An URL for retrieving the status of the payout. |

If a HTTP status code 422 error occurs, an array of Error objects are returned.

**Errors**

| HTTP status code | Description |
|---|---|
| 201 Created | The Payout request was successfully created. Will return a Location header. |
| 400 Bad Request | The Create Payout Request operation was malformed. |
| 401 Unauthorized | There are authentication problems with the certificate. Or the Swish number in the certificate is not enrolled. Will return nothing else. |
| 422 Unprocessable Entity | There are validation errors. Will return an array of Error objects. |
| 500 Internal Server Error | There was some unknown/unforeseen error that occurred on the server, this should normally not happen. Will return nothing else |

**Validation errors**

The following validation errors might be returned when the HTTP status code is 422.

| Error code | Description |
|---|---|
| PA01 | Invalid format of a field or otherwise invalid information in request. |
| PA02 | Invalid format of 'amount' |
| AM03 | Invalid format of 'currency' |
| RP02 | Invalid format of 'message' |
| FF08 | Invalid or missing 'payerPaymentReference' |
| ACMT13 | Bank does not support 'PAYOUT' |
| ACMT14 | Payer is not allowed to perform 'PAYOUT' |
| ACMT15 | Payee is not allowed to receive 'PAYOUT' |
| TM01 | Swish system timed out |

| | |
|---|---|
| RF07 | Transaction could not be executed |

**Example request**

```
curl -v --data '
{"payload":{"signingCertificateSerialNumber":"7d70445ec8ef4d1e3a713427e97
3d097","amount":"200","payoutInstructionUUID":"D7799FA730C4460EBB8EAE7121
F6FA3B","payoutType":"PAYOUT","payerPaymentReference":"orderId","instruct
ionDate":"2020-03-
23T16:08:49","payerAlias":"1234679304","currency":"SEK","payeeAlias":"467
68648198","message":"message","payeeSSN":"196210123235"},"signature":"Qhe
1pwq0SrveqHx9+dhSEvfQ4UQ3fdwWAcrGjJ4n+5wqu4MPhaGvV+30UBWSbJ3GeQUGdrObtDxX
Qe0XnciKOBYosW3Xn3FiZiPSPsPgG9FpFb1TkqIdaZkeNhiIcKe8KxfRyoFFr2b3FCUkORUuD
RE+0Nk11Uo0iaaDd/FPtmWkCkvnAfMxO1/d+RDVxIDEf8dLFh3Q7XGffXcwezdqO6sYUtUIZB
z4s3iCJdzzf0GeudOg+rCYn6E4UGAr8JEhc7ijaFF4PlUJR4kvDeOqgJyW8fdoTg/o2EO4jDI
skbl20xRra5VH+VkrT8HrVCr7wd8nkBP+LhzSVas5LqR1kQzhl6jA09Y+UA23EnhgxOIR5cHd
myo/blun7+qwFTCS3bC4U/hgRpf5CiYF87p0Ebi0/r8k7WBaMOl7l5j06ag566oQxpX1qcWDf
m+fxZZfI1cMO8YcjVFT6E8kC7bNmRqAQGptsoZF103azHrn/uuPafRb+aXx1l4VrUfqmFCL4o
PfMyxEQu+VM8U9IvruBgCKSyELZIzkGldxpgQZdYQb46czcVcLMUb/CuQpyudwUejWazgFOp/
TBCNo8W6ff6+W+9Uz42a6CsSY0Bxm0qwL0GS2GDW9ARkjeLZ5J57R5XTlm1aMlc15zVg8dYLF
dJrUpHWDnjc16iBuaRXe31Q=","callbackUrl":"https://mystore/payments/swish/c
allback/"}' -H "Content-Type: application/json" POST
https://mss.cpc.getswish.net/swish-cpcapi/api/v1/payouts/ --cert
"Swish_Merchant_TestCertificate_1234679304.p12:swish" --cert-type p12 --
cacert "Swish_TLS_RootCA.pem"
```

**Example response**

```
HTTP/1.1 201
Location: https://mss.cpc.getswish.net/cpc-
swish/api/v1/payouts/D7799FA730C4460EBB8EAE7121F6FA3B
Content-Length: 0
Date: Mon, 23 Mar 2020 15:08:50 GMT
Connection: keep-alive
```

### 5.4.7 Retrieve Payout

**Request**

```
GET /api/v1/payouts/{id}
```

**Parameters**

| Name | Description |
|---|---|
| id | The identifier of the payout to retrieve.<br>Example: 11A86BE70EA346E4B1C39C874173F088 |

**Request body**

None

**Response**

If successful, a Payout object is returned.

If a HTTP status code 422 error occurs, an array of Error objects are returned.

**Errors**

| HTTP status code | Description |
|---|---|
| 200 OK | The Payout is returned. |
| 400 Bad Request | Request failed for some reason, i.e. bad payoutReference.<br>Details provided in the error message in the body. |
| 401 Unauthorized | There are authentication problems with the certificate. Or the Swish number in the certificate is not enrolled. |
| 500 Internal Server Error | A server error has occurred.<br>Details provided in the error message in the body. |

**Validation errors**

The following validation errors might be returned when the HTTP status code is 422.

| Error code | Description |
|---|---|
| PA01 | Invalid format of a field or otherwise invalid information in request. |
| TM01 | Swish system timed out |
| RF07 | Transaction could not be executed |
| UNKN | Unknown error. |

**Example request**

```
curl -v "Content-Type: application/json" GET
https://mss.cpc.getswish.net/swish-
cpcapi/api/v1/payouts/FE3082B0B38844C797E7499EADACCDF7 --cert
"Swish_Merchant_TestCertificate_1234679304.p12:swish" --cert-type p12 --
cacert "Swish_TLS_RootCA.pem"
```

**Example response**

```
< HTTP/1.1 200
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Mon, 23 Mar 2020 15:23:08 GMT
<
* Connection #1 to host mss.cpc.getswish.net left intact
{"paymentReference":"43DA7306F8DA426D8D7F82C939721031","payoutInstruction
UUID":"FE3082B0B38844C797E7499EADACCDF7","payerPaymentReference":"orderId
","callbackUrl":"https://mystore/payments/swish/callback/","payerAlias":"
1234679304","payeeAlias":"46768648198","payeeSSN":"196210123235","amount"
:200.00,"currency":"SEK","message":"message","payoutType":"PAYOUT","statu
s":"PAID","dateCreated":"2020-03-23T15:17:29.016Z","datePaid":"2020-03-
23T15:17:33.016Z","errorMessage":null,"additionalInformation":null,"error
Code":null}
```

## 5.5 API Objects

### 5.5.1 Date format

The date fields use the **YYYY-MM-DDThh:mm:ss.sssTZD** date format. Times are returned in the UTC time zone.

### 5.5.2 Payment Request

**Properties**

| Property | Type | Description |
|---|---|---|
| id | string | Payment request ID. |
| payeePaymentReference | string | Payment reference of the payee, which is the merchant that receives the payment. This reference could be order id or similar. Allowed characters are a-z A-Z 0-9 -_.+*/ and length must be between 1 and 36 characters. |
| paymentReference | string | Payment reference, from the bank, of the payment that occurred based on the Payment request. Only available if status is PAID. |
| callbackUrl | string | URL that Swish will use to notify caller about the outcome of the Payment request. The URL has to use HTTPS. |
| payerAlias | string | The registered cellphone number of the person that makes the payment. It can only contain numbers and has to be at least 8 and at most 15 numbers. It also needs to match the following format in order to be found in Swish: country code + cellphone number (without leading zero). E.g.: 46712345678 |
| payeeAlias | string | The Swish number of the payee. |
| amount | number | The amount of money to pay. The amount cannot be less than 1 SEK and not more than 999999999999.99 SEK. Valid value has to be all numbers or with 2-digit decimal separated by a period. |
| currency | string | The currency to use. The only currently supported value is SEK. |
| message | string | Merchant supplied message about the payment/order. Max 50 chars. Allowed characters are the letters a-ö, A-Ö, the numbers 0-9 and the special characters :;.,?!()”. |
| status | string | The status of the transaction. Possible values: CREATED, PAID, DECLINED, ERROR. |
| dateCreated | string | The time and date that the payment request was created. |
| datePaid | string | The time and date that the payment request was paid. Only applicable if status was PAID. |

| Property | Type | Description |
|---|---|---|
| errorCode | string | A code indicating what type of error occurred. Only applicable if status is ERROR. |
| errorMessage | string | A descriptive error message (in English) indicating what type of error occurred. Only applicable if status is ERROR. |
| additionalInformation | string | Additional information about the error. Only applicable if status is ERROR. |

**Error codes**

| errorCode | Description |
|---|---|
| ACMT03 | Payer not enrolled. |
| ACMT01 | Counterpart is not activated |
| ACMT07 | Payee not enrolled. |
| RF07 | Transaction declined. The payment was unfortunately declined. A reason for the decline could be that the payer has exceeded their defined Swish limit. Please advise the payer to check with their bank. |
| BANKIDCL | Payer cancelled BankID signing. |
| FF10 | Bank system processing error. |
| TM01 | Swish timed out before the payment was started. |
| DS24 | Swish timed out waiting for an answer from the banks after payment was started. Note: If this happens Swish has no knowledge of whether the payment was successful or not. The merchant should inform its consumer about this and recommend them to check with their bank about the status of this payment. |
| BANKIDONGOING | BankID already in use. |
| BANKIDUNKN | BankID is not able to authorize the payment. |

### 5.5.3 Refund

**Properties**

| Property | Type | Description |
|---|---|---|
| id | string | Refund ID. |
| payerPaymentReference | string | Payment reference supplied by the merchant. This could be order id or similar. |

| Property | Type | Description |
|---|---|---|
| originalPaymentReference | string | Reference of the original payment that this refund is for. |
| paymentReference | string | Reference of the refund payment that occurred based on the created refund. Only available if status is PAID. |
| callbackUrl | string | URL that Swish will use to notify caller about the outcome of the refund. The URL has to use HTTPS. |
| payerAlias | string | The Swish number of the merchant that makes the refund payment. |
| payeeAlias | string | The cellphone number of the person that receives the refund payment. |
| amount | number | The amount of money to refund. The amount cannot be less than 1 SEK and not more than 999999999999.99 SEK. Moreover, the amount cannot exceed the remaining amount of the original payment that the refund is for. |
| currency | string | The currency to use. The only currently supported value is SEK. |
| message | string | Merchant supplied message about the refund. Max 50 chars. Allowed characters are the letters a-ö, A-Ö, the numbers 0-9 and the special characters :;.,?!()”. |
| status | string | The status of the refund transaction. Possible values:<br>• VALIDATED - Refund ongoing<br>• DEBITED - Money has been withdrawn from your account<br>• PAID - The payment was successful<br>• ERROR - An error occurred. See list of error codes for all potential error conditions. |
| dateCreated | string | The time and date that the payment refund was created. |
| datePaid | string | The time and date that the payment refund was paid. |
| errorCode | string | A code indicating what type of error occurred. Only applicable if status is ERROR. |
| errorMessage | string | A descriptive error message (in English) indicating what type of error occurred. Only applicable if status is ERROR |
| additionalInformation | string | Additional information about the error. Only applicable if status is ERROR. |

**Error codes**

| errorCode | Description |
|---|---|
| ACMT07 | Payee not enrolled. |
| ACMT01 | Counterpart is not activated. |
| RF07 | Transaction declined. Please contact your bank. |
| FF10 | Bank system processing error. |
| DS24 | Swish timed out waiting for an answer from the bank after payment was started. Note: If this happens Swish has no knowledge of whether the payment was successful or not. The merchant should inform its consumer about this and recommend them to check with their bank about the status of this payment. |

### 5.5.4 Create Payout Request

**Properties**

| Property | Type | Format | Description |
|---|---|---|---|
| payload | object | A Payout object. | |
| callbackUrl | string | https://<host[:port]>/...<br><br>Max length: 265 characters | URL that swish system will use to notify caller about the result of the payment request. The URL has to use HTTPS.<br>If not set it is the responsibility of the caller to check the status of the payout request using a GET operation. |
| signature | string | Base64 encoded. Max length 265 characters. | Signature of the hashed payload. |

### 5.5.5 Payout

**Properties**

| Property | Type | Format | Description |
|---|---|---|---|
| payoutInstructionUUID | string | UUID - 32 hexadecimal (16-based) digits. | An identifier created by the merchant to uniquely identify a payout instruction sent to the Swish system. Swish uses this identifier to guarantee the uniqueness of a payout instruction and prevent occurrence of unintended double payments. |
| payerPaymentReference | string | 1-35 characters. Valid characters are: a-zA-Z0-9-_.+*/ | Merchant specific reference. This reference could be order id or similar. |

| Property | Type | Format | Description |
|---|---|---|---|
| payerAlias | string | Numeric, 10 digits | The merchant Swish number that makes the payment. |
| payeeAlias | string | Numeric, 8-15 digits | The Swish number of the payee.<br>No preceding “+” or zeros should be added. It should always be started with country code. |
| payeeSSN | string | YYYYMMDDnnnn | 12 digit SSN of the payee. Will be validated against the enrolled SSN of the payee. |
| amount | number | 100.00 | Amount to be paid. Only period/dot (”.”) are accepted as decimal character with maximum 2 digits after. Digits after separator are optional. |
| currency | string | SEK | The currency to use. The only currently supported value is SEK. |
| payoutType | string | PAYOUT | Currently only “PAYOUT” is allowed – meaning immediate payout. |
| message | string | Alphanumeric,<br>0-50 chars. | Custom message. |
| status | string | CREATED, INITIATED, BIR_PAYMENT_INITIATED, DEBITED, PAID, ERROR. | The status of the payout request. |
| signingCertificateSerialNumber | string | Serial number of the certificate in hexadecimal format (without the leading ‘0x’). Max length 64 digits. | The public key of the certificate will be used to verify the signature |
| instructionDate | string | YYYY-MM-DDThh:mm:ssTZD | The time and date that the payout request was created. |
| dateCreated | string | YYYY-MM-DDThh:mm:ssTZD | The time and date that the payout request was created. |
| datePaid | string | | The time and date that the payout request was paid. Only applicable if status was PAID. |
| errorCode | string | | A code indicating what type of error occurred. Only applicable if status is ERROR. |
| errorMessage | string | | A descriptive error message (in English) indicating what type of error occurred. Only applicable if status is ERROR. |

| Property | Type | Format | Description |
|---|---|---|---|
| additionalInformation | string | | Additional information about the error. Only applicable if status is ERROR. |

### 5.5.6 Operation

**Properties**

| Property | Type | Description |
|---|---|---|
| op | string | The operation to perform. Possible values: “replace” |
| path | string | Document path. |
| value | string | The new value. Possible values: “cancelled” |

### 5.5.7 Error

**Properties**

| Property | Type | Description |
|---|---|---|
| errorCode | string | A code indicating what type of error occurred. |
| errorMessage | string | A descriptive error message (in English) indicating what type of error occurred. |
| additionalInformation | string | Additional information about the error. |

# 6. Support

## 6.1 Deployment support

Please see the manuals and FAQ available at https://developer.getswish.se/.

If you can't find the technical information you need, you can contact the deployment support organization. The email address is: tekniksupport@getswish.se.

For all commercial questions, please contact your bank.

## 6.2 Operational status information

Operational status information is available at https://www.getswish.se/driftinformation/.